CSE474

Assignment 03

Name: Md. Farhadul Islam

ID: 19201086

Representing Model Uncertainty in Deep Learning using Monte Carlo Dropout

Abstract. Dropout is a technique for preventing overfitting in a deep learning model. At each update of the training phase, dropout functions by setting the outgoing neurons that make up hidden layers to 0. In the field of applied machine learning, deep learning technologies have gotten a lot of attention. However, such regression and classification algorithms do not account for model uncertainty. Bayesian models, on the other hand, provide a theoretically based framework for reasoning about model uncertainty, but they generally come at a high computing expense. The Monte Carlo (MC) dropout technique provides a scalable way to learn a predictive distribution. In MC dropout, the dropout is applied at both training and test time unlike the normal dropout. The prediction is no longer deterministic at test time, but rather depends on the nodes you pick at random. As a result, the model may predict different values each time given the same datapoint.

Keywords: Deep Learning · Neural Network · Monte Carlo Dropout

1 Introduction

In statistics, the aphorism "all models are wrong" is typically extended to "all models are wrong, but some are useful".This aphorism is a famous quote often attributed to the British statistician George E. P. Box. The aphorism acknowledges that statistical or scientific models will always fall short of reality's intricacies, yet they can nevertheless be useful. In deep learning, our predicted outputs are not deterministic. The stochastic nature of deep learning makes the model uncertain. Knowing the uncertainty of a model is very crucial when it comes to decision making. When dealing with medical, financial data the uncertainty can cause serious complications if the uncertainty of a deep learning model is not represented.

Deep learning models usually lack the representation of uncertainty and make overconfident and faulty predictions [3,4].

Bayesian techniques like Bayesian neural networks (BNNs) and Gaussian processes have gotten a lot of press for providing uncertainty measurements in addition to predicting probabilities. BNNs and Gaussian processes, unlike single predictions, produce predictive distributions in which BNN weights are combined with priors distribution [2], whereas Gaussian processes include priors over functions [5]. These methods are effective theoretically, but tough to do in practice.

Gal and Ghahramani [1] have proposed a very simple and easy to implement method for representing model uncertainty. They have introduced MC dropout, which is basically normal dropouts used in the testing phase of the model. My project will use this method to represent uncertainty in deep learning models.

2 Literature Review

There are several studies on representing uncertainty in deep learning models. Such as Bayesian Methods, SGD Based Approximations, Methods for Calibration of DNNs.

2.1 Bayesian Methods

In Bayesian model averaging, a distribution is placed over model parameters, and then these parameters are marginalized to build a full predictive distribution. The foundational studies of Neal [2] and MacKay [8] established Bayesian approaches as the state-of-the-art approach to learning using neural networks in the late 1990s. Modern neural networks, on the other hand, frequently include millions of parameters, and the posterior over these parameters (and hence the loss surface) is very non-convex, necessitating mini-batch techniques to get to a suitable solution space [9]. Bayesian techniques have been essentially intractable for current neural networks as a result of these factors.

Markov chain Monte Carlo (MCMC): MCMC was very popular for inference with neural networks, through the Hamiltonian Monte Carlo (HMC) work of Neal [2]. However, HMC requires full gradients, which is computationally intractable for modern neural networks. Chen et al. [10] presented stochastic gradient HMC (SGHMC), which allows stochastic gradients to be employed in Bayesian inference, which is important for both scalability and exploring a space of solutions with high generalization. In the stochastic gradient situation, stochastic gradient Langevin dynamics (SGLD) [11] employs first order Langevin dynamics. In the case of indefinitely tiny step sizes, both SGHMC and SGLD asymptotically sample from the posterior. Using limited learning rates causes approximation problems in reality, and tweaking stochastic gradient MCMC algorithms can be tricky.

Variational Inference: Graves [5] proposed that the weights of neural networks be fitted with a Gaussian variational posterior approximation. The reparameterization strategy was suggested by Kingma and Welling [12] for training deep latent variable models, and numerous variational inference techniques based on the reparameterization trick were developed for DNNs. While variational approaches function well on modestly sized networks, they are challenging to train on bigger designs such as deep residual networks, according to other studies.

Dropout Variational Inference: Gal et al. [1] propose a novel theoretical framework that casts deep neural network (NN) dropout training as approximate Bayesian inference in deep Gaussian processes. As a direct outcome of this theory, we may use dropout NNs to represent uncertainty, retrieving information from current models that has previously been discarded. This solves the challenge of conveying uncertainty in deep learning without losing test accuracy or computational complexity. This approach is very easy to implement and used widely.

SWA-Gaussian for Bayesian Deep Learning: Maddox et al. [14] propose SWA-Gaussian (SWAG) for Bayesian model averaging and uncertainty. Stochastic Weight Averaging (SWA), which iterates with a modified learning rate schedule and computes the first moment of stochastic gradient descent (SGD), has recently been proven to increase generalization in deep learning. With SWAG they form an approximate posterior distribution over neural network weights by fitting a Gaussian using the SWA solution as the first moment and a low rank plus diagonal covariance also derived from the SGD iterates. Then, they sample from this Gaussian distribution to perform Bayesian model averaging. In line with studies defining the stationary distribution of SGD iterates, their experiments show that SWAG approximates the form of the real posterior estimation.

2.2 SGD Based Approximations

Mandt et al. [17] proposed to use the iterates of averaged SGD as an MCMC sampler, after analyzing the dynamics of SGD using tools from stochastic calculus. Chen et al. [19] investigate the problem of statistical inference of true model parameters based on SGD when the population loss function is strongly convex and satisfies certain smoothness conditions.

2.3 Methods for Calibration of DNNs

For improved calibration, Lakshminarayanan et al. [20] proposed utilizing ensembles of many networks, as well as an adversarial loss function to be utilized when possible. Outside of probabilistic neural networks, Guo et al. [3] developed temperature scaling, a process that rescales the logits of DNN outputs for improved calibration using a validation set and a single hyperparameter. Using a similar rescaling method, Kuleshov et al. [21] offer calibrated regression. These methods are quite tougher to implement comparatively.

3 Methodology and Experimentation

3.1 Dataset

Gal et al. [1] uses several datasets for several tasks. For representing uncertainty in regression task, they use Mauna Loa CO2 concentrations dataset [15]. For their classification task MNIST handwritten digit dataset [16] is used. A training set of 60,000 samples and a test set of 10,000 examples are available in this dataset. The digits have been size-normalized and centered in a fixed-size image.

In my project, I will be using multiple datasets for this task as well, including both classification and regression task. For the regression task I will run my experiments on the Boston Housing Dataset [6,7]. The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. I will use MNIST [16] or a Brain Tumor Classification dataset [18].

3.2 Frameworks and Github Repositories

We can do deep learning tasks using several programming languages, such as Python 3 [27], R [29], MATLAB [28]. There are many machine learning and deep learning frameworks for python, which are being used regularly worldwide. Scikit-Learn [25] and Tensorflow [22] are popular for machine learning tasks. Both of them have a rich data processing tool-set as well. Keras [23] is an extension of Tensorflow (Version 2), which is used for deep learning tasks. It also has a rich tool-set of data processing. Another popular framework is PyTorch [24], which is built for deep learning tasks. In my project, I will be using Tensorflow/Keras for deep learning tasks. I will be needing scikit-learn for data processing tasks. Other frameworks and libraries will also be used.

MC dropout and its variants are implemented in [38]. CIFAR-10 dataset is used in this project. The whole project was programmed using PyTorch. The Lenet architecture [26] is developed for this task. The results show that the MC dropout achieves the highest accuracy.

In [39], the author estimate uncertainty in CNN classification of dogs and cats images using monte carlo dropout. Author represents the uncertainty of the model with scatter plots. The CNN model developed in this project is comparatively bigger and more complicated. It has 4 convolutional layers and ends with a fully connected layer.

In [40] we see a practical implementation of [42]. Gal et al. [42] introduce an efficient Bayesian convnet that outperforms previous techniques in terms of resilience to over-fitting on limited data. This is accomplished by applying a probability distribution to the kernels of the convnet. The practical implementation can be seen in this project, where the author developed Lenet using Keras.

The practical implementation of [43] is given in [41]. They used Keras to implement their idea. They propose a novel powerful fusion model named UncertaintyFuseNet that consists of Ensemble Monte Carlo (EMC) dropout. The results show that their fusion model can achieve the optimum performance using both experimented datasets. The repository does not provide the whole model, but it has all the data processing and visualization tasks required. This repository is helpful as I might use a medical dataset in my project.

3.3 Method Architecture

Gal et al. [1] interpret dropout as a sampling method that is equivalent to a variational approximation of a deep Gaussian process. A deep Gaussian process is a Bayesian machine learning model that ordinarily outputs a probability distribution, and conventional dropout at test time may be used to determine properties of this underlying distribution. The estimated variance of the distribution is used to determine the model's uncertainty for a given input. Monte Carlo dropout is the name for this approach of estimating uncertainty. A neural network is initially trained regularly using conventional dropout before being used to implement Monte Carlo dropout. The network is run T times with conventional dropout, all with the same input but different randomly generated dropout masks each time, to perform inference on an input sample.

3.4 Experimental Analysis

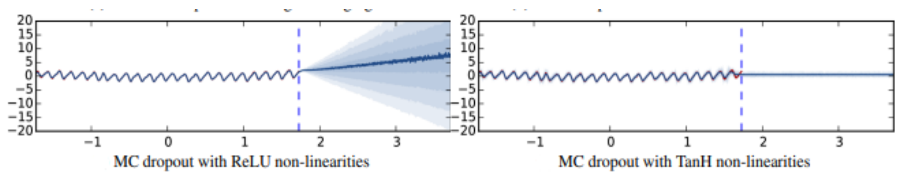

Fig. 1: Regression Task Analysis of Using MC Dropout [1]

Gal et al. [1] run experiments on of the properties of the uncertainty estimates obtained from dropout NNs and convolutional networks on the tasks of regression and classification. They compare the uncertainty obtained from different methods and show that model uncertainty is important for classification tasks.

In Fig. 1, the uncertainty is increasing far from the data for the ReLU model, whereas for the TanH model it stays bounded. The first graph of Fig. 1 gets more scattered and spread proving the high uncertainty of the model. This indicates the model's uncertainty is in ReLU.
In Fig. 2 the plots show the softmax input value and softmax output value for the 3 digits with the largest values for each corresponding input. The model predicts the matching class when the softmax input for a class is greater than the softmax input for all other classes (class 1 for the first 5 images, class 5 for the next 2 images, and class 7 for the rest in figure 2, first plot). If the uncertainty envelope of a class differs significantly from those of other classes (for example, the leftmost picture), the input is categorized with high confidence. However, if the uncertainty envelope overlaps that of other classes, the softmax output uncertainty can be as large as the entire space, even though the softmax output

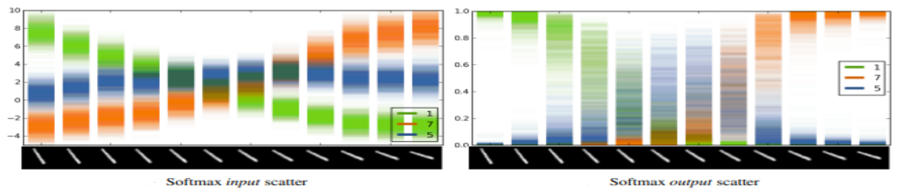

Fig. 2: Classification Task Analysis of Using MC Dropout [1]

can be arbitrarily high. This indicates the model's uncertainty in its softmax output value (in the prediction).

Seoh et al. [30] have done a qualitative analysis of MC dropout. They focus on the effects of dropout rates and model precision, the number of epochs and improvements in predictive performance. The research shows us how dropout rate has a great impact on the model and how it fails to catch the uncertainty of a model. Unlike dropout rate, model precision does not seem to change the overall trends of uncertainty captured from their experiment. The number of training epochs is important to catch the uncertainty. The more you increase the number of training epochs the better you capture the uncertainty. But, we have to consider that, the model can overfit with a very high number of training epochs. Their analysis revealed that MC dropout reduced not just RMSE error, but also variation across splits. Given that we only had a tiny quantity of data to work with, this was extremely promising. This might indicate that, even with limited data, we can generate more reliable predictions using NN. However, it was unclear if there were any performance benefits compared to regular dropout. However, whether there are any performance improvements over normal dropout was unclear. In Fig. 3, the boxplots of RMSE scores across 10 train-test splits for MC dropout and standard dropout, done on a 1-layer FC NN and Boston Housing dataset (506 rows). In no case did it appear that MC dropouts had lower mistake rates or a more consistent range of results across splits. We also looked at additional datasets and found the same patterns.

Lemay et al. [31] demonstrated that MC sampling during test time leads to more reliable classification models providing more stable and repeatable predictions on different images from the same patient. When using MC sampling, only regression models did not exhibit a consistent improvement. MC sampling is versatile since it can be used to any model type and architecture and is simple to implement. According to their findings, utilizing MC models for binary, multi-class, and ordinal models for all tasks enhanced repeatability. When MC iterations were added to regression models for various medical applications, no significant increase in classification or repeatability was seen.

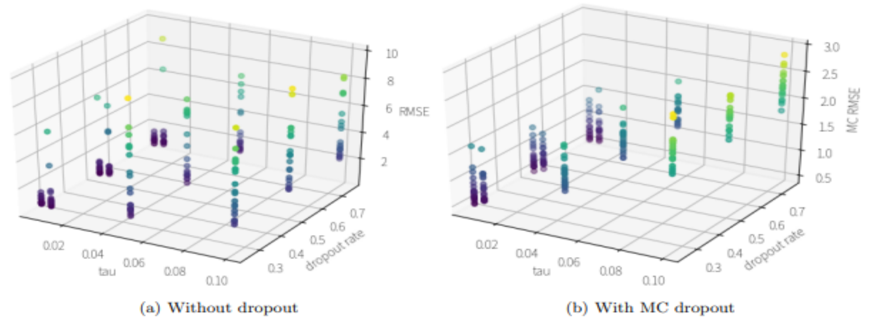

Fig. 3: Prediction Analysis with and without MC Dropout [30]

4 Applications and Further Improvements

MC dropout can be used in many areas of deep learning. Many applications of Monte Carlo dropout have been identified in practice, including time-series prediction [33] and medical imaging [34]. Bayesian neural networks [32] and ensemble-based techniques [20] are two further ways suggested to quantify model uncertainty. Miok et al. [35] proposed a method of generating data with MC dropout. The idea is to incorporate Monte Carlo Dropout method within Autoencoder (MCD-AE) and Variational Autoencoder (MCD-VAE) as efficient generators of synthetic data sets.

MC dropout is very easy to implement but it has some flaws. Osband et al. [36] discusses the risks regarding MC-dropout. Risk is described as a model's intrinsic variability, whereas uncertainty seems to encapsulate our unsure conviction in the forecast value. Consider the well-known example of a coin flip. The chance of drawing head or tail carries considerable risk. In the case of a biased coin, however, our views are updated as new observations are collected, which is referred to as the uncertainty. When dealing with feature vectors around the decision boundary in binary classification tasks, this "risk" phrase is appropriate. As a result, MC-dropout generates flipped outputs for highly close classes. Additionally, the test points that occur between two overlapping classes reveal aleatoric uncertainty. As a result, the terms risk and aleatoric uncertainty are somewhat interchangeable. The model risk and aleatoric uncertainty of the predictions are irreducible in both forms. This example will be demonstrated later in this paper's experiment.

Shamsi et al. [37] combine cross entropy with Expected Calibration Error (ECE) and Predictive Entropy to present two novel loss functions (PE). The obtained findings clearly illustrate that using the new suggested loss functions, a calibrated MC-Dropout approach may be obtained. The new hybrid loss functions had a big impact on reducing the overlap between the distributions of uncertainty estimates for accurate and wrong predictions without affecting the model's overall performance. This improves the MC dropout by providing better performance.

5 Conclusion

This technique of representing uncertainty using MC dropout is very simple and very useful. Dropouts are typically used to avoid model overfitting but this technique also allows us to represent the uncertainty of the model. MC sampling is flexible since it can be used to any model type and architecture and is simple to implement. Moreover, MC dropout effects in terms of model repeatability. Monte Carlo dropout also offers a wide range of practical applications. Bayesian neural networks and ensemble-based techniques are two more ways proposed to quantify model uncertainty. The benefit of Monte Carlo dropout over these methods is that no modifications to the model training procedure are required, whereas both of these alternatives result in a significant increase in training complexity.

References

1. Gal, Yarin & Ghahramani, Zoubin. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. Proceedings of The 33rd International Conference on Machine Learning.
2. Neal RM (2012) Bayesian learning for neural networks, vol 118. Springer, Berlin.
3. Guo, Chuan & Pleiss, Geoff & Sun, Yu & Weinberger, Kilian. (2017). On Calibration of Modern Neural Networks.
4. Kendall, Alex & Gal, Yarin. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?.
5. Graves, A. (2011). Practical variational inference for neural networks. In Advances in neural information processing systems, pages 2348–2356.
6. Harrison, D. and Rubinfeld, D.L. (1978) Hedonic prices and the demand for clean air. J. Environ. Economics and Management 5, 81–102.
7. Belsley D.A., Kuh, E. and Welsch, R.E. (1980) Regression Diagnostics. Identifying Influential Data and Sources of Collinearity. New York: Wiley.
8. MacKay, David J. C. (1992). Bayesian Interpolation. Neural Computation, 4(3), 415–447. doi:10.1162/neco.1992.4.3.415
9. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In International Conference on Learning Representations. arXiv: 1609.04836.
10. Chen, T., Fox, E. B., and Guestrin, C. (2014). Stochastic Gradient Hamiltonian Monte Carlo. In International Conference on Machine Learning. arXiv: 1402.4102.
11. Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 681–688.
12. Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In International Conference on Learning Representations.
13. MacKay, David J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. Neural Computation, 4(3), 448–472. doi:10.1162/neco.1992.4.3.448

14. Maddox, Wesley & Garipov, Timur & Izmailov, Pavel & Vetrov, Dmitry & Wilson, Andrew. (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning.
15. Dr. Pieter Tans, NOAA/GML (gml.noaa.gov/ccgg/trends/) and Dr. Ralph Keeling, Scripps Institution of Oceanography (scrippsco2.ucsd.edu/).
16. LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database.
17. Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic Gradient Descent as Approximate Bayesian Inference. JMLR, 18:1–35.
18. Sartaj. (2019). Brain Tumor Classification (MRI), 2. Retrieved from https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri.
19. Chen, X., Lee, J. D., Tong, X. T., and Zhang, Y. (2016). Statistical Inference for Model Parameters in Stochastic Gradient Descent. arXiv: 1610.08637.
20. Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Advances in Neural Information Processing Systems.
21. Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate Uncertainties for Deep Learning Using Calibrated Regression. In International Conference on Machine Learning, page 9.
22. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
23. Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras
24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
25. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.
26. LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". Neural Computation. 1 (4): 541–551. doi:10.1162/neco.1989.1.4.541. ISSN 0899-7667. S2CID 41312633
27. Van Rossum, G., Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.
28. MATLAB. (2010). version 7.10.0 (R2010a). Natick, Massachusetts: The MathWorks Inc.
29. R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.
30. Seoh, Ronald. (2020). Qualitative Analysis of Monte Carlo Dropout.

31. Lemay, Andreanne Hoebel, Katharina Bridge, Christopher Egemen, Didem Rodriguez, Ana Cecilia Schiffman, Mark Campbell, John Kalpathy-Cramer, Jayashree. (2021). Monte Carlo dropout increases model repeatability.
32. Y. Gal, “Uncertainty in deep learning,” Ph.D. dissertation, University of Cambridge, 2016.
33. L. Zhu and N. Laptev, “Deep and confident prediction for time series at Uber,” arXiv preprint arXiv:1709.01907, 2017.
34. A. Jungo, R. McKinley, R. Meier, U. Knecht, L. Vera, J. Perez-Beteta, D. Molina-Garc ´ ´ıa, V. M. Perez-Garc ´ ´ıa, R. Wiest, and M. Reyes, “Towards uncertainty-assisted brain tumor segmentation and survival prediction,” in Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, A. Crimi, S. Bakas, H. Kuijf, B. Menze, and M. Reyes, Eds. Cham: Springer International Publishing, 2018, pp. 474–485.
35. Miok, Kristian Nguyen Doan, Dong Zaharie, Daniela Robnik-Sikonja, Marko. (2019). Generating Data using Monte Carlo Dropout.
36. Osband, I. (2016). Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout. Workshop on Bayesian Deep Learning, NIPS .
37. Shamsi, Afshar Asgharnezhad, Hamzeh Abdar, Moloud Tajally, AmirReza Khosravi, Abbas Nahavandi, Saeid Leung, Henry. (2021). Improving MC-Dropout Uncertainty Estimates with Calibration Error-based Optimization.
38. sungyubkim, Monte-Carlo Dropout and its variants, (2019), GitHub repository, https://github.com/chezou/solar-power-prediction
39. statsu1990, Uncertainty estimation in deep learning using monte carlo dropout with keras, (2019), GitHub repository, https://github.com/chezou/solar-power-prediction
40. homaralex, Implementation of (parts of) the experiment on MNIST from Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference, (2020), GitHub repository, https://github.com/homaralex/mc-dropout-mnist
41. moloud1987, UncertaintyFuseNet: Robust Uncertainty-aware Hierarchical Feature Fusion with Ensemble Monte Carlo Dropout for COVID-19 Detection, (2021), GitHub repository, https://github.com/moloud1987/UncertaintyFuseNet-for-COVID-19-Classification
42. Gal, Yarin Ghahramani, Zoubin. (2015). Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference.
43. Abdar, M., Salari, S., Qahremani, S., Lam, H., Karray, F., Hussain, S., Khosravi, A., Acharya, U.R., Nahavandi, S. (2021). UncertaintyFuseNet: Robust Uncertainty-aware Hierarchical Feature Fusion with Ensemble Monte Carlo Dropout for COVID-19 Detection. ArXiv, abs/2105.08590.